From A to B, randomly: a point-to-point random trajectory generator for animal movement

Technitis, Georgios ; Othman, Walied ; Safi, Kamran ; Weibel, Robert

Abstract: For applications in animal movement, we propose a random trajectory generator (RTG) algorithm that combines the concepts of random walks, space-time prisms, and the Brownian bridge movement model and is capable of efficiently generating random trajectories between a given origin and a destination point, with the least directional bias possible. Since we provide both a planar and a spherical version of the algorithm, it is suitable for simulating trajectories ranging from the local scale up to the (inter-) continental scale, as exemplified by the movement of migrating birds. The algorithm accounts for physical limitations, including maximum speed and maximum movement time, and provides the user with either single or multiple trajectories as a result. Single trajectories generated by the RTG algorithm can be used as a null model to test hypotheses about movement stimuli, while the multiple trajectories can be used to create a probability density surface akin to Brownian bridges.

DOI: https://doi.org/10.1080/13658816.2014.999682

International Journal of Geographical Information Science
Vol. xx, No. xx, Month 2014, 1–24

RESEARCH ARTICLE

From A to B, randomly: A point-to-point random trajectory generator for animal movement

Georgios Technitis[a*], Walied Othman[a], Kamran Safi[b,c] and Robert Weibel[a]

[a]*Department of Geography, University of Zurich, Winterthurerstrasse 190, 8057 Zurich, Switzerland*; [b]*Department of Migration and Immuno-ecology, Max Planck Institute for Ornithology, Am Obstberg 1, 78315 Radolfzell, Germany*; [c]*Department of Biology, University of Konstanz, Konstanz, Germany*

(*received August 2014*)

For applications in animal movement, we propose a random trajectory generator (RTG) algorithm that combines concepts of random walks, space-time prisms, and the Brownian bridges movement model and is capable of efficiently generating random trajectories between a given origin and a destination point, with the least directional bias possible. Since we provide both a planar and a spherical version of the algorithm, it is suitable for simulating trajectories ranging from the local scale up to the (inter-)continental scale, as exemplified by the movement of migrating birds. The algorithm accounts for physical limitations, including maximum speed and maximum movement time, and provides the user with either single or multiple trajectories as a result. Single trajectories generated by the RTG algorithm can be used as a null model to test hypotheses about movement stimuli, while the multiple trajectories can be used to create a probability density surface akin to Brownian bridges.

Keywords: Movement simulation; movement ecology; random walk; Brownian bridges; space-time prisms.

*Corresponding author. Email: georgios.technitis@geo.uzh.ch

ISSN: 1365-8816 print/ISSN 1362-3087 online

DOI: 10.1080/1365881YYxxxxxxxx
http://www.informaworld.com

1. Introduction

The past decade has seen the advent of affordable and ubiquitous technologies that allow tracking moving objects of various kinds at increasing levels of accuracy. Despite these unprecedented opportunities to generate large quantities of real trajectory data, there are several applications where the simulation of trajectories remains a necessity. Examples include completing missing data in recorded trajectories (Wentz *et al.* 2003); the generation of synthetic patterns in trajectories as a basis for performance testing of spatiotemporal database management systems (Brinkhoff 2002) or pattern recognition algorithms (Pelekis *et al.* 2013); and the simulation of particular moving objects such as cars (Joubert *et al.* 2010) or pedestrians (Torrens *et al.* 2012).

In animal ecology, which forms the application focus of this paper, movement simulation takes a special position and has to respond to particular requirements, reviewed in the following section. Among others, sampling densities in animal tracking are typically much lower than in the tracking of humans or human-made objects. Furthermore, the knowledge about the animals being tracked is typically incomplete; indeed, the very purpose of movement tracking is to find out more about the animals' behaviour. Thus, the movement models used to simulate animal trajectories are typically of probabilistic nature, such as various forms of random walks (Codling *et al.* 2008, 2010) or Brownian bridges (Bullard 1991, Benhamou 2011). However, as we will show below, these approaches have limitations.

The contribution of this paper is thus a random trajectory generator (RTG) algorithm that overcomes the limitations of existing movement models. It is based on a combination of concepts of random walks, space-time prisms, and Brownian bridges, and allows to efficiently and reliably generate either individual or large numbers of trajectories between two given endpoints, with randomly placed intermediate points, subject to a given time budget and speed regime. Since we provide versions of the algorithm in both planar and spherical geometry, it is capable of generating long-haul trajectories such as those typical of bird migration. As will be experimentally shown, this algorithm also offers an alternative to Brownian bridges (Horne *et al.* 2007) in generating empirical probability distributions of movement between two endpoints. It thus also extends over the recent work by Song and Miller (2014) by offering a tool to generate individual trajectories rather than generating visit probability distributions.

We start off the remainder of this article by a discussion, in Section 2, of the role and the requirements of movement simulation in animal ecology. In Section 3 we review the relevant literature and identify the limitations of current methods. In Section 4 the basic, planar geometry version of our RTG algorithm is presented, while in Section 5 we explain the extensions necessary to account for the spherical shape of the Earth. Section 6 presents and discusses several experiments, with the aim of evaluating the proposed algorithm. Section 7 offers conclusions and an outlook on future research.

2. Movement simulation in animal ecology

Research in animal ecology aims to understand the decisions that animals take balancing between their environmental context and their physiological needs and capacities. The overall aim is to understand the causes and consequences of animal movement, which crucially determine the ecological role of a species and, via differential survival of the individuals, the evolutionary processes that ultimately are the origin of the nat-

ural diversity on which coexistence in species communities is founded. Scientists are interested in understanding resource selection, which is influenced by and influences the spatio-environmental and physiological limitations in the movement repertoire. The recent decline in global species diversity, as a consequence of human activities, has made it increasingly urgent to understand, for instance, the human-wildlife conflict as a consequence of large scale transnational movements and growing human population. Likewise, effects on migratory corridors need to be studied, for example effects induced by land use change, increasing hunting pressure, or mortality through obstacles such as power lines or wind energy farms.

One of the major contributions of simulation, in movement ecology, is making more with less data. Ethical considerations about animal welfare as well as logistic limitations make repeated recapture and frequent handling of wild animals most often unfeasible. There is a tradeoff between limited battery capacity and the limitations of the weight that the animals can be expected to carry with minimal effect on their natural behaviour (usually considered max 5% of their body mass). High sampling frequencies and long observation periods mean either large batteries or recapturing of animals to replace batteries. If animals are tracked in the wild, temporal sampling rates are typically low. It is thus common not to know exactly where an individual has been between two consecutive GPS fixes, because the two points may be temporally far apart. Given the time budget available between the two fixes and some reasonable assumption about the movement speed, the animal may potentially have roamed large areas between two measured fixes. However, although the true location of an animal cannot be derived from the available tracking data, we can develop a movement model using statistical assumptions to simulate where the animals *might* have been.

An additional contribution of simulation, in movement ecology, is the creation of null models (Nathan *et al.* 2008). Null models, in ecology, "entertain the possibility that nothing has happened, that a process has not occurred, or that change has not been produced by a cause or interest" (Esa 1982). Comparing the null model against the respective observed data helps to test hypotheses relative to why an animal followed a specific route. Quantitatively, a null distribution provides an expectation of an unbiased route choice, whereas the distribution of the real data incorporates the individual's preference in relation to, for instance, its context. By repeatedly comparing the two distributions against each other and testing the same hypothesis on multiple datasets, movement ecologists eventually infer rules about animal navigation (Biro *et al.* 2007, Lohmann *et al.* 2004) and space utilisation (Block *et al.* 2011, Börger *et al.* 2008); behavioural analysts formalise knowledge on foraging strategies (Bartumeus *et al.* 2010, Humphries *et al.* 2010); and epidemiologists trace the potential origin of a bird flu outbreak (Elveback *et al.* 1976).

Finally, it is crucial to point out the necessity of simulating individual trajectories, and not falling back on statistical distributions such as Brownian or Levy motion. By generating individual trajectories, the researcher has the ability to identify, focus on and analyse interesting patterns in movement data, without being affected by the shortcomings of a-priori assumptions about distributions.

Requirements. From the above, we can summarise the requirements for a trajectory generator. First, the procedure has to be capable of generating trajectories between two specific points, such as two consecutive fixes along the flight path of a migrating bird, sampled at one fix per day, and potentially several hundreds of kilometres apart. Second, the points of the simulated trajectory must be randomly placed in space, to allow generating the null hypothesis. Third, the generated trajectories must be realistic, i.e. it

must be possible to condition them by parameters such as a given time budget, minimum and maximum speed, etc. Fourth, the algorithm must be efficient, so that large numbers of trajectories can be generated between the two given endpoints, allowing the generation of empirical probability surfaces. Finally, since the endpoints may be separated by large distances, the algorithm must take into account the spherical shape of the Earth.

3. Related work

3.1. *Random Walks*

Random walks (RW) are a fundamental concept originally developed to describe the irregular motion of pollen particles suspended in water, also known as Brownian motion (Codling *et al.* 2008). The concept has been extended and adapted to meet the needs of movement ecology. Specifically, the correlated random walk (CRW)(Benhamou 2006) and the drifted random walk (DRW)(Codling *et al.* 2008, 2010) have been used to forecast the value of a temporal variable, such as location, in ecology.

The drifted random walk expresses a global directional tendency, or drift, in the trajectory. It would thus seem like a possible candidate for our problem of generating a random trajectory that can be controlled to meet a given endpoint. However, since the DRW approach only ties the trajectory down to a single point (the starting point), meeting the endpoint can only be achieved by increasing the drift in the random walk (thus reducing path randomness), by coarse discretization (thus reducing the degree of realism), or by shooting many paths in a trial-and-error process (terribly inefficient). Furthermore, DRW has no way of observing a given time limit.

A blend of biased (i.e. drifted) and correlated random walk (BCRW) may be used in an attempt to combine local persistence and global bias along the walk (Codling *et al.* 2008), however, without fundamentally improving the underlying individual methods. Further attempts to model complex movement behaviour and spatial heterogeneity using Bayesian Markov Chain Monte Carlo methods and bivariate Ornstein-Uhlenbeck processes (Harris and Blackwell 2013) prove to be useful for accounting for changes in an animal's behaviour over time. However, that particular study refers to areas of movement, and not to a point-to-point trajectory generation algorithm.

3.2. *Brownian bridges*

While the random walk methods are modelling trajectories emanating from a single starting point, the Brownian bridges (BB) approach attempts to tackle the problem of connecting an origin and a destination, accounting for the total time of travel. The BB movement model was introduced to ecology to estimate the home range of animals (Bullard 1991). The task of BB was to model an animal's space utilisation distribution, that is, create a map depicting the probability of utilisation for each location in a given study area.

An intuitive description of the underlying mathematical model (Horne *et al.* 2007, Perony *et al.* 2012), is that a particle (moving object) is released on a bridge connecting the two locations (Horne *et al.* 2007, Perony *et al.* 2012), with a specific time budget. This creates a constant, fluctuating within a range, drift towards the end point. The movement behaviour in the model is considered homogeneous and the focus lies on generating a visit probability surface, rather than individual trajectories.

3.3. *Space-time prism*

Based on Hägerstrand's theory of Time Geography, the concept of the space-time prism (STP) allows defining the set of all points that can be reached by an individual given a maximum possible speed from a starting point in space-time and an ending point in space-time (Hägerstrand 1970) in the 3-D space of the space-time cube, denoted by the potential path space (PPS), or its 2-D projection to geographic space, the potential path area (PPA)(Miller 1991). These basic concepts have been extended to model and measure accessibility along networks (Miller 1999), to incorporate uncertainty in the definition of the origin and destination (Kuijpers *et al.* 2010), or as an analytical tool to solve the *alibi query* (Kuijpers *et al.* 2011). Very recently, Song and Miller (2014) have presented the combination of the STP and Brownian bridges, whereby the BB model was used to generate visit probability distributions that were truncated to the PPS by the use of the STP model. This approach has the definite advantage of truncating the visit probabilities of the BB to the PPS, generating zero probabilities in areas that cannot possibly be reached. However, it does not allow to generate individual trajectories, which is particularly important if varying behavioural parameters should be modelled and no analytical solution exists to compute the truncated visit probabilities over large geographic distances, that is, on a sphere.

3.4. *Research challenges*

As follows from the above review, none of the existing methods is capable of meeting all the requirements listed in Section 1.

CRW and DRW may be useful from the point of view of ecological modelling, but suffer from a lack of computational efficiency when the destination is a point rather than an area of attraction (i.e. the hit-to-miss ratio is very poor).

The result of the Brownian bridges model is a probability surface that may be regarded as representative of reality in terms of the space utilisation of a moving object. However, once an individual trajectory is extracted from a BB probability surface, using for instance conditional Brownian walk, the result will be a highly unrealistic mode of movement that is biologically unjustified for animals. Furthermore, generating Brownian bridges is computationally expensive and thus in practice not feasible to compute large numbers of trajectories.

Space-time prisms offer an interesting concept of generating an envelope of accessibility in space and time, that is, the potential path space (PPS). With the recent work by Song and Miller (2014), an approach has been presented of truncating the occupancy probabilities of Brownian bridges to the more realistic domain of the PPS (and PPA in 2-D space). However, it still inherits some of the weaknesses of BBs: lacking facility of generating individual trajectories with varying movement parameters; high computational load; and no analytical solution for spherical coordinates.

In order to address these challenges, we thus propose an algorithm that

- takes the drifted random walk model of movement as a basis;
- offers significantly improved efficiency over the DRW model, by tying the generated walk to both the origin *and* the destination;
- provides results as single/multiple trajectories;
- provides space utilisation maps similar to those generated by Brownian bridges; and
- limits the creation of trajectories to the PPA defined by the space-time prism.

4. Planar version of the algorithm

4.1. *Underlying modelling framework*

The conceptual model of movement simulation underlying our algorithm consists of three main steps (Technitis and Weibel 2012):

(1) initialise the conditions of the simulation;
(2) identify possible area for the next move; and
(3) determine the object's next movement with the highest probability based on a specified level of confidence.

In Step 1, the conditions defining the simulation are specified. In our case, that implies defining the coordinates of the origin A and the destination B of the movement simulation; the available time budget to move from A to B; the temporal sampling interval used to generate fixes along the trajectory; and the maximum speed that the object is allowed to move. Step 2 represents the focus of this paper. The proposed algorithm generates realisations of trajectories within the possible area of movement, equivalent to the PPA of the STC approach, using the control parameters of Step 1. Step 3 is only implicitly dealt with in this paper. By generating sufficient number of trajectories an empirical occupancy probability distribution is approximated (cf. Section 6.2).

For our simulation, we assume the following conditions: The moving object performs a drifted random walk at equal time intervals per step and with a fixed maximum speed, in a continuous and homogeneous space (2-D Euclidean space in this section; spherical geometry in Section 5).

4.2. *The algorithm*

4.2.1. Computing the Potential point Area (PpA)

The trajectory to be generated will consist of a number of intermediate points, separated by the temporal sampling interval Δt. For the creation of each of them, we start by constructing the potential point area, or short *PpA*. We use a lower-case middle 'p' to differentiate from the space-time prism's term Potential Path Area (PPA). The PpA then denotes the area in the plane where an intermediate point can be possibly placed. The following example describes this step.

Example 1. We assume that the distance $\overline{AB}$ between the origin A and destination B of the movement is 300 km. Point A is the origin, where the moving object departs at $t_0 = 0$, and point B is the destination, where the object arrives at $t_f = 5$ hours later. The task, then, is to create a trajectory from A to B, given a maximum speed of the moving object $V_{max} = 70$ km/h, within the time budget $t_{total} = 5$ hours, and given a sampling interval $\Delta t = 1$ h.

If the moving object starts at A and travels for one hour as the sampling interval Δt commands, the area where the object would be expected to be is equivalent to a circle of radius r_A, where r_A is the maximum distance that can be covered in Δt given V_{max}, in our case $r_A = 70$ km. At the same time, B needs to be reached at $t_f = 5$ h. Hence, one hour before $t_4 = 4$ h, the moving object ought to be within its maximum speed radius from B in order to be able to reach the destination in the final hour remaining. Likewise, the moving object, 4 hours before it reaches point B, should be within a distance that can be covered within these 4 hours, that is, within a circle around B of radius $r_B = 4 \times V_{max}$

= 280 km. Consequently the PpA after one hour is defined by the intersection of the two circles from A and B, respectively (Fig. 1). Next, we will explain how random placement of an intermediate point location within the PpA is incorporated in the procedure.

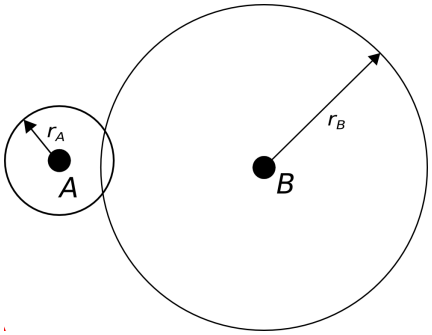

Figure 1.: The Potential point Area (PpA) after Δt is defined by the intersection of a circle of radius $r_A = V_{max} = 70$ km around A, and a circle of radius $r_B = 4V_{max} = 280$ km around B.

4.2.2. Placing points randomly within the PpA

Now that the PpA is defined, we need a specific location to actually place the new point. Since one of the requirements is to generate a random trajectory, the location needs to be selected randomly within the PpA. The point placement is performed in four steps. Firstly, the PpA is calculated using the intersection of the two circles around A and B (cf. Fig. 1). As a second step, the minimum bounding rectangle MBR_1 around the PpA is created (shaded in Fig. 2). This MBR serves as an approximate candidate region to speed up the following two steps.

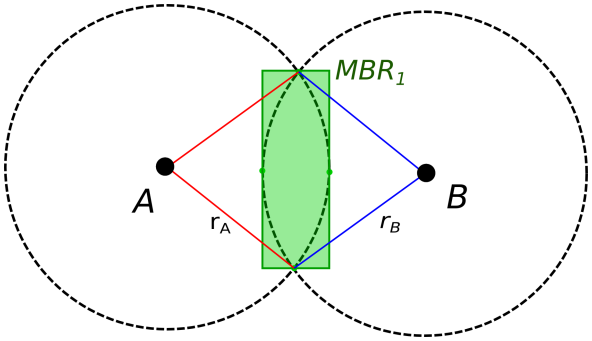

Figure 2.: Creating the minimum bounding rectangle MBR_1 of the PpA.

In the third step, a potential point $P_{pot}(x_{pot}, y_{pot})$ is temporarily placed within MBR_1. This is performed by drawing two random coordinates (x_{pot}, y_{pot}) from two uniform distributions with range $[MBR_{X_{min}}, MBR_{X_{max}}]$ and $[MBR_{Y_{min}}, MBR_{Y_{max}}]$, respectively. Finally, in the fourth step, we test whether P_{pot} actually lies within the PpA. If the distance d_A is less than radius r_A, while at the same time the distance d_B is less than the radius r_B, then P_{pot} is accepted, and treated as the origin for the creation of the next point. Otherwise, P_{pot} lies inside MBR_1 but falls outside the PpA, thus the third and fourth step of the procedure are repeated until P_{pot} can be accepted.

Using the MBR to generate and test P_{pot} greatly simplifies computations and increases efficiency, although this approach will also generate some misses (cf. 6.1). Furthermore,

compared to an approach that requires sampling of the PpA by some sampling scheme such as a grid, in our approach the resulting point locations and the computational performance of the algorithm are independent of the resolution of the sampling scheme. Overall, this approach preserves the randomness of the path, while observing given constraints (i.e. time budget and maximum speed), efficiently connecting the starting point with the destination point.

4.2.3. Completing the trajectory

Once the point P_{pot} is accepted, it becomes the new origin A_1 to initiate the creation of the next intermediate point A_2 of the trajectory. The time budget remaining for the completion of the trajectory is now decreased by Δt. The following example describes the steps necessary to complete the trajectory.

Example 2. The sampling interval is $\Delta t = 1$ h, hence $r_A = V_{max} = 70$ km. However, since the moving object has already travelled for one hour to generate the first point, the time to destination B is now $t_{total} - 2 \times \Delta t = 3$ h, and $r_B = 3 \times V_{max} = 210$ km. With these new values for r_A and r_B, the procedure described in Sections 4.2.1 and 4.2.2 is repeated to create the point A_2 (Fig. 3), and analogously for the remaining points A_3 and A_4 (Fig. 4).

Ultimately this procedure yields a trajectory that consists of the sequence of points $\{A, A_1, A_2, A_3, A_4, B\}$, as shown in Figure 4. This sequence represents a random but feasible trajectory with a given origin and destination, while not exceeding the available time budget and the given maximum speed.

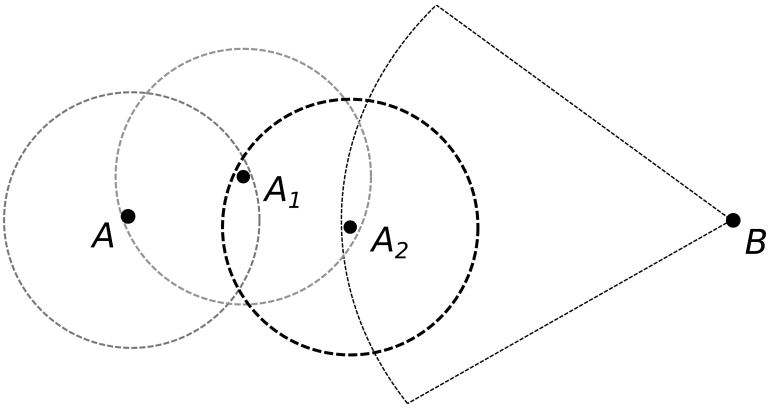

Figure 3.: Repeating the point generation procedure

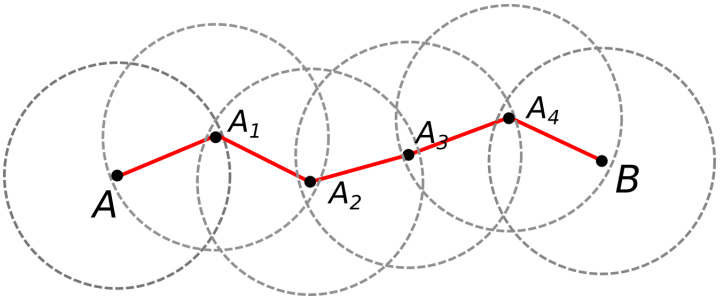

Figure 4.: Final trajectory

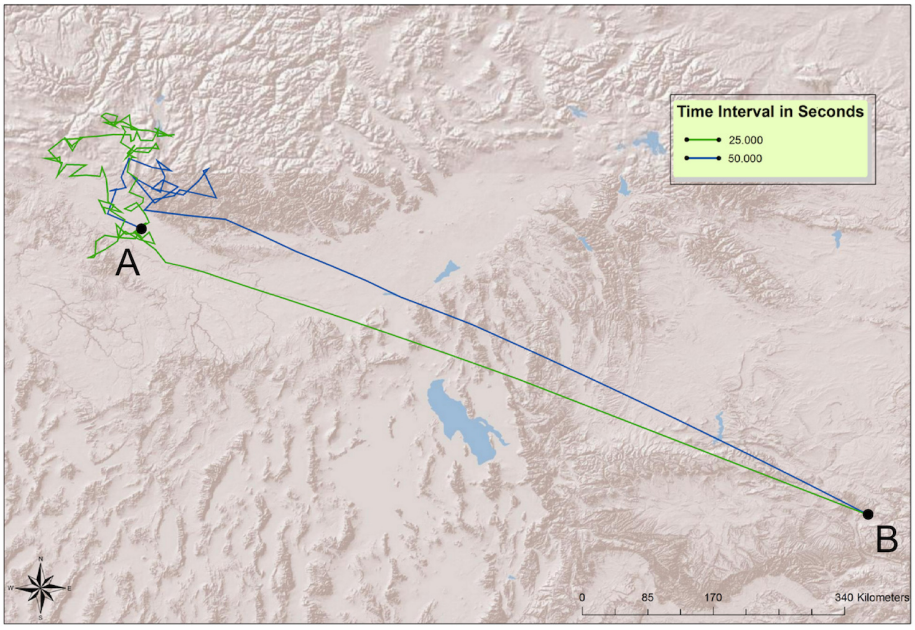

Figure 5.: Two example trajectories that meet the requirements of Section 1 but are unrealistic, since most of distance between the origin and destination is covered by a heavily drifted walk.

4.2.4. Limitation in the degrees of freedom

After creating each point, the time budget is updated based on the need to complete the remaining trajectory, thus imposing a growing limitation on the PpA. The two factors affecting the placement of a new point are the last created point and the temporal sampling interval Δt (Fig. 4). The change of origin affects the position of the circle of all possible locations for the next point to be created (the circle around A_i), while the destination and the decreasing available time budget narrows down the extent of the second circle (the circle around B). In other words, the algorithm as described so far will initially generate largely Brownian motion until the time budget is almost used up, forcing the remaining few time-steps to be spent on 'hurrying' towards the destination to reach it on time. Figure 5 shows two example trajectories that both exhibit the bipartite movement regime—near-random vs. heavily drifted, respectively—which is the consequence of the basic algorithm. It also becomes apparent that the coarser time interval of 50,000 seconds allows for less vertices to be placed, thus reducing the extent of the free movement part. It is worth pointing out that this very ability to scale the temporal step of the random walk differentiates RTG from Brownian motion, where by definition Δt ought to converge towards 0.

In order to avoid the effect of 'wandering' excessively during the initial part of the trajectory generation, the reduction of the degrees of freedom in the PpA should be distributed randomly over the trajectory. This equalised temporal distribution can be achieved by making the order of the point placement no longer sequential. The approach used here is to choose an order of point placements that is random in the time-steps.

The algorithm proceeds by first creating a list of time-steps necessary for generating the trajectory. Out of this list, each time-step is randomly selected, one after another, for the placement of the corresponding point. This is illustrated in Figure 6, where a set

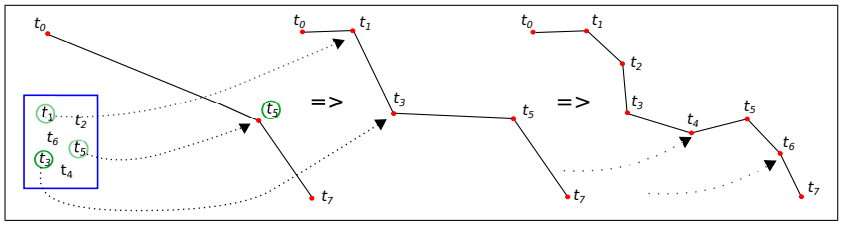

Figure 6.: Selecting random time-step and creating the corresponding point

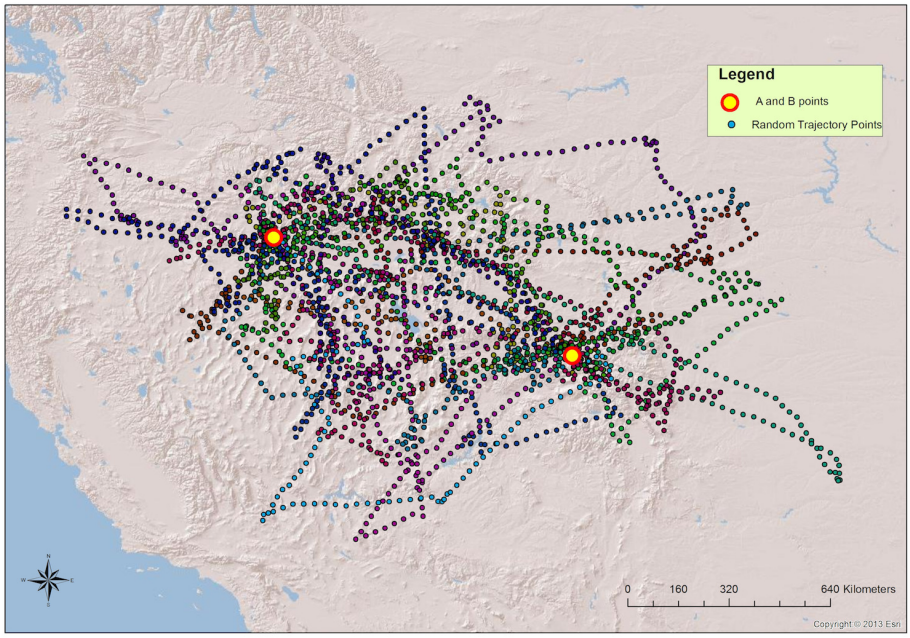

Figure 7.: A set of 20 trajectories generated using the improved algorithm with random temporal order of point placement. The set up of the experiment is identical to the one visualised in figure 5.

of six intermediate points $\{t_1, t_2, ..., t_6\}$ needs to be generated in order to complete the trajectory AB. Instead of the sequential ordering $(t_1, t_2, ...t_6)$ used initially, we create the points in random sequence, for instance: $(t_5, t_1, t_3, t_6, t_4, t_2)$. Each time, the last created point becomes the new starting point for the algorithm while the destination is set to the temporally next available point - if applicable.

Once the list is empty, all the necessary points have been created, and the trajectory is compiled by sorting them in temporally sequential order. A set of trajectories generated by this procedure are depicted in Figure 7. We notice that the bipartite pattern of Figure 5 is no longer visible; the limitation in the degree of randomness of the walks is now evenly distributed along the trajectories.

5. Spherical version of the algorithm

Owing to the spherical shape of the Earth, using the planar version of the algorithm to create trajectories over large spatial scales bears the problem of spherical distortion, violating the coherence of distance, angles and areas in non-Euclidean space. For the

reconstruction of large scale displacements, such as continental or intercontinental migratory movement, it is therefore necessary to extend the algorithm to accommodate spherical geometry. It is important to note here that implementing the model on the sphere does not imply the third dimension of topography. In the present version of the algorithm, the implementation does therefore not account for elevation.

In our algorithm we compute a circle-circle intersection and approximate it with a bounding box or speeding up random placement of points. In the spherical version of the algorithm these circles are circles on a sphere (the Earth). If we ignore the spherical shape of the Earth, this implies that the longitude λ and the latitude φ of geographical coordinates are equated with the x and y of a planar coordinate system: $x = \lambda, y = \varphi$. This is equivalent to using an equirectangular (or plate carrée) projection, which is known to cause massive distortions towards the poles. Using such a projected map to define circles on the sphere, the projected circles will therefore not retain the properties of a circle. Thus, calculating the bounding box of their intersection at arbitrary positions on the globe would be rather difficult. However, we know that the distortions of projected circles are minimal along the equator. We therefore transpose the two circles to the Equator, one of them on the intersection with Central Meridian, keeping their relative distance the same. Thus, the distortion of the projected circles is minimised, and the bounding box around the intersection can be expressed by a minimum and maximum value for the width and height. This enables us to efficiently select a random point from the bounding box. Once this point is placed, we verify, using the Haversine formula whether or not the point actually is contained in the circle-circle intersection.

Circle-circle intersections on the sphere, apart from the different geometry, could be comprised of the same cases seen in Euclidian space. The two circles might be too far apart and have no intersection. One of the circles might be contained in the other, without their boundaries intersecting. The two circles might be touching in a single point, or finally, they might have a proper intersection with two distinct intersection points. The latter case is what we will explore in detail; the other cases are easily accounted for by evaluating inequalities on their radii and distance between the centres.

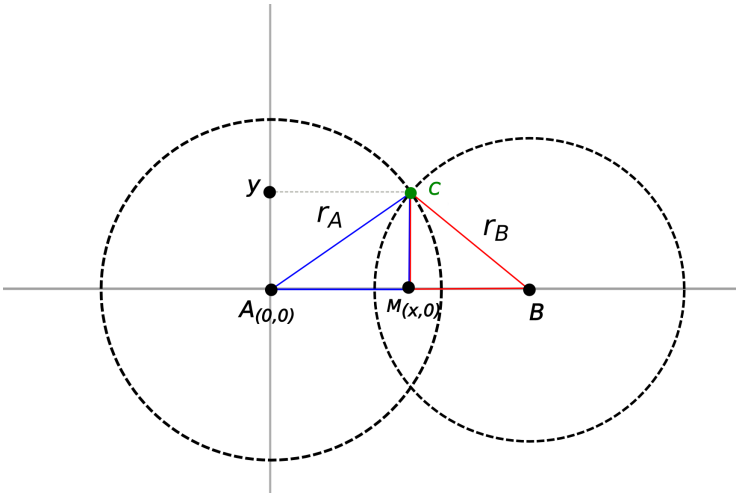

Figure 8.: Circle-circle intersection on the sphere

Figure 8 depicts two circles with a proper intersection. Although it contains straight lines, Figure 8 is to be interpreted as lying on the sphere, and not in the plane. In this setting, the known variables are the circles' radii, and the distance between their

centres. The unknown variables we wish to compute are the coordinates of the intersection points, specifically, the coordinates of C. The other intersection point shares the same x coordinate, but has a mirrored (i.e. negative) y coordinate.

In 2-D Euclidean space we would solve a set of Pythagorean equations for the right triangles $\triangle AMC$ and $\triangle BMC$. For right triangles on a sphere, we use the spherical analogue of Pythagoras' theorem and this returns the following identities:

$$\begin{aligned} \cos\left(\frac{r_A}{R_E}\right) &= \cos\left(\frac{x}{R_E}\right) \times \cos\left(\frac{y}{R_E}\right) \\ \cos\left(\frac{r_B}{R_E}\right) &= \cos\left(\frac{\overline{AB}-x}{R_E}\right) \times \cos\left(\frac{y}{R_E}\right) \end{aligned} \tag{1}$$

where r_A stands for the radius of the circle with centre A, r_B for the radius of circle with centre B, and R_E for the radius of the Earth. The distance between the two circles is given by $\overline{AB}$. The unknowns we wish to solve for, are x and y, the coordinates of the upper intersection point.

These equations are solved following a standard procedure. An inverse cosine is only determined up to its sine, and a cosine will change its sign when π is added to its argument. So, once we compute one solution of x, we have to account for three others: $-x$, $x+\pi$ and $-x+\pi$. We restrict the circles' radii to that of less than half the Earth's circumference and can thus rule out the latter two solutions. By verifying with the set of equations, we can show that unless $x = 0$, either x or $-x$ is a solution.

We are, in fact, not interested in the value of x, but rather in the value of y, as it determines (half) the height of the bounding box. The width of the bounding box depends entirely on the circles' radii and the distance between their centres. Figure 9 depicts this bounding box.

The width of the bounding box, defined by x_{min} and x_{max}, respectively, depends entirely on the circles' radii and the distance between their centres (Fig. 9), as expressed by the following equations

$$\begin{aligned} x_{min} &= \overline{AB} - r_B \\ x_{max} &= r_A. \end{aligned} \tag{2}$$

The height of the bounding box, defined by y_{min} and y_{max}, is obtained by solving the system of equations 1 for y. Now that we have the limits of the bounding box, we repeatedly generate a point therein by generating random coordinates within these limits. We then verify its distance to the two circle centres using the Haversine formula and accept it if it is also contained in the circle-circle intersection. Otherwise we generate a new point within the bounding box and repeat as described in Section 4.2.4). Finally, points that have been found to lie within the corresponding circle-circle intersections (i.e. the spherical PpA) are transformed back to their actual location on the globe.

6. Evaluation

In order to assess the performance of the proposed RTG algorithm, we proceeded in two steps, involving the verification (internal evaluation) and the validation (external evaluation) of the algorithm. The verification step (6.1) involved testing the computational

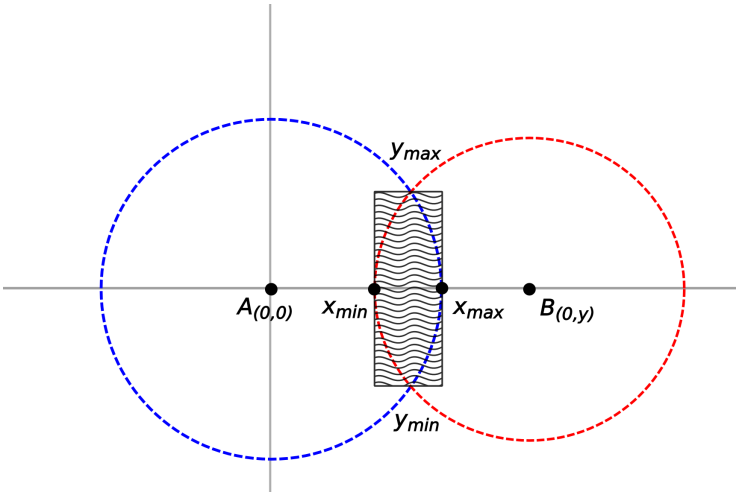

Figure 9.: The bounding box coordinates

performance as well as the robustness of the algorithm. In the validation step (6.2) some points were omitted from the input trajectory and then used to assess the accuracy of the simulation. The same inputs were also used with the BB approach for comparison. An overall discussion (6.3) concludes this section.

6.1. *Verification*

Performance Analysis. The purpose of this analysis was to test the robustness and efficiency of the algorithm under different parameter combinations. In order to obtain more robust estimates, runtimes were calculated as an average over five generated sample runs. The model was tested against five main parameters: the total number of trajectories created, the number of point within each trajectory, the maximum speed of the moving object, the geographic distance covered, and the total duration of movement.

Table 1 summarises the empirical analysis of the computational efficiency, realised on a standard desktop PC (Intel QuadCore i7 CPU 870 @ 2.93 GHz, with 8 GB RAM, running LinuxMint 16 (petra) 3.11.0-12-generic and Python 2.7). Each scenario is separated by a horizontal line altering one or two parameters at a time, leaving the rest unchanged. It becomes obvious that the RTG algorithm maintains a linear relation between the running time and each parameter tested. As expected the increase in the number of points, the greater distance, the total number of trajectories, or the duration of the simulated movement, simply increase the load and not the complexity of the calculations.

As described above the proposed algorithm first approximates the PpA with a bounding box to speed up the point placement step (4.2.2). The price to pay is that misses might occur: points that lie inside the MBR but fall outside the PpA. In the implementation of the algorithm we limited the maximum number of unsuccessful attempts to place a random point to 10; otherwise, a miss is reported for the particular point. This has happened rarely, however, as Table 2 shows. What is reported as accuracy of the algorithm is the percentage of these misses (i.e. the random points created falling outside the PpA) of the total number of points to be created. For example, in the first line the algorithm failed to create 66 points of 50,000 requested, yielding an accuracy of 99.868 %.

Table 1.: Empirical analysis of computational efficiency

Run	# of Trajs	# of Points	Speed (m/s)	Distance (km)	Duration of Movement(s)	Average Runtime(s)
1	5	45	0.4198	2.386	21645	0.478
2	50	45	0.4198	2.386	21645	4.728
3	500	45	0.4198	2.386	21645	47.05
4	5	88	0.4198	2.386	21645	0.89
5	50	88	0.4198	2.386	21645	9.016
6	500	88	0.4198	2.386	21645	90.124
7	10	150	0.4198	2.386	21645	2.909
8	10	218	0.4198	2.386	21645	4.37
9	10	272	0.4198	2.386	21645	5.411
10	10	362	0.4198	2.386	21645	7.205
11	10	410	0.4198	2.386	21645	8.233
12	10	505	0.4198	2.386	21645	10.287
13	1	520	0.4198	0.02	373419	1.135
14	1	1053	0.4198	0.02	373419	2.258

Table 2.: Robustness of the algorithm

Misses	Trajectories	Points	Distance(km)	Accuracy
66	500	100	200	99.868%
12	100	100	200	99.88%
27	500	100	2	99.946%
2	100	100	2	99.98%

Boundary Cases. Both in the temporal and in the spatial dimension the RTG algorithm may encounter boundary cases, that is, cases when the lower or upper bound of an input parameter is reached. The lower temporal bound can be described by the minimum time required to reach the destination. In this case the moving object has to follow the shortest possible path in order to make it to the destination in time. Thus, the trajectory will form the straight line connecting the two endpoints. This case is handled by creating all the necessary points on the straight line connecting the origin and destination points, as this is the only feasible solution.

The lower spatial bound is reached when the intersection area between the two circles (see Fig. 1) is so small that it may be considered negligible for the purpose of the simulation, that is, if it falls below the minimum spatial resolution. As can be seen in Figure 10(i), when the candidate area is larger than the threshold (in this case 1 m for the width of the bounding box), the point is placed randomly as outlined in Section 4.2.2. If, however, the candidate area is smaller than the threshold (Fig.10(ii)), the centroid of the intersection area is selected as the new point.

The opposite case, an upper bound, is reached in the case of long trajectories, such as those covering half or even the entire globe. These represent potential challenges for a trajectory generator. Thus, to gain a better overview of the algorithm's robustness, we tested such cases and also special areas (i.e. areas around the poles). For this test we needed a leading trajectory, that can be created with any movement model or parameters for as long as it covers distances created than 80-90°. In our case, we created a 1000-step correlated random walk starting from the equator, moving up to the North Pole. More specifically we used the wrapped-Cauchy function of the circular package in the R statistics language, with mean direction of the distribution set to 0, and the concentration

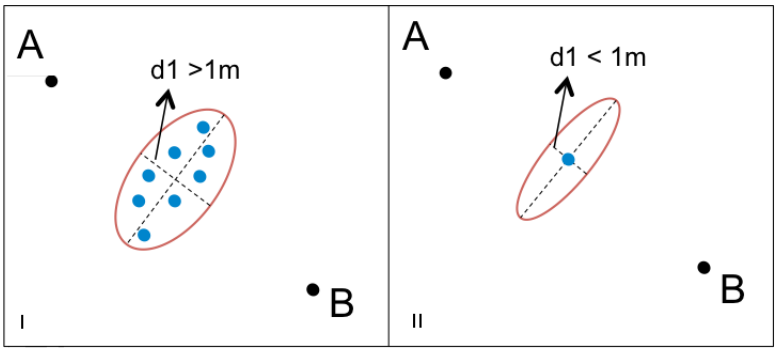

Figure 10.: Boundary cases of the spatial resolution in point placement

parameter randomly set to 0.8. Once the 1000 steps were calculated, the trajectory was down-sampled keeping 1 point out of 10, and the RTG algorithm was executed 100 times for each pair of remaining consecutive points. This resulted in 100 trajectories, generating points at the same timestamps as the missing points. Figure 11 shows the result of this operation. The green points denote the CRW that was then down-sampled and used as input to the RTG algorithm. Each of the 100 simulated trajectories has a randomly assigned colour, with 9 new points created for each pair of consecutive sample points. Owing to the equirectangular projection used for the main map of Figure 11, trajectories are particularly distorted towards the North Pole. Thus, the inset map shows all RTG runs on a spherical projection.

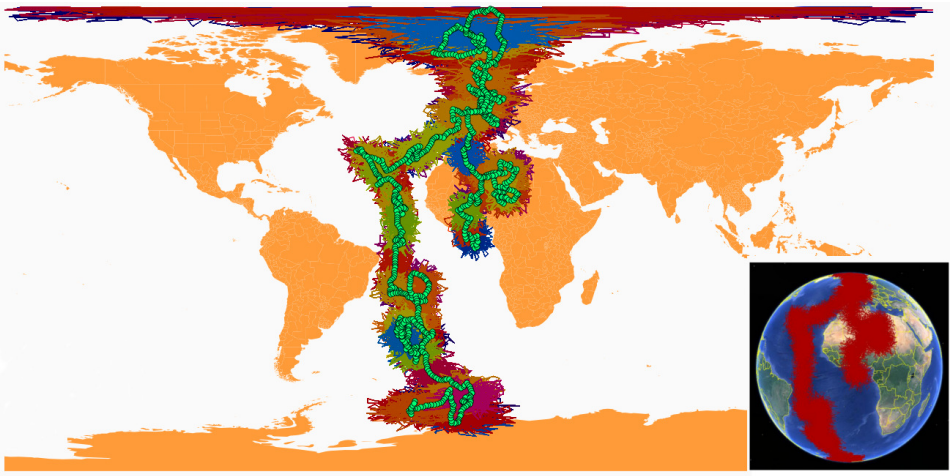

Figure 11.: 100 runs of the RTG algorithm on a 1000-step, resampled CRW (green points) that starts from the intersection of the equator with the prime meridian, moving towards the North Pole and then back to the southern hemisphere again. The inset map shows all RTG runs in a spherical projection.

6.2. *Validation*

Procedure. The data used for our validation experiment was taken from the Movebank site (see Acknowledgements). The group working on this dataset, set up an experiment tracking a flock of white storks that follows a migration path from Poland to South

Africa, via Sudan and Tanzania. The birds were tagged with GPS devices, recording the position in two modes: a high frequency mode (average time interval = 49 mins) for maximum spatiotemporal resolution and a lower frequency mode (average time interval = 360 mins) for making sure that the battery will last until the end of the journey. The stork trajectory in Figure 12 clearly depicts the two different sampling modes. This particular trajectory was used for our evaluation scenario. In a different application context, the RTG algorithm would be used to realistically densify the coarse part of the data, making it comparable to the high-frequency part. For the purposes of evaluation though, we focused on the high-frequency part of the trajectory. The initial dataset was down-sampled, omitting 50 % of the points, and then it was used as input data for simulating the omitted points at $\Delta t/2$, as one point will be created between each pair of points. We ran the simulation 50 times for each missing part of the trajectory and then rasterised the result, yielding the count of trajectories in each pixel, thus creating a density surface. The raster template used was the same pixel size (1118.113 x 1498.257 m) and extent with the one BB created, for easier comparison between the results. These counts can be transformed into relative densities as a proportion of the total trajectories, and after normalisation considered as a probability surface that can be compared against the probability surface extracted from the BB, with probability values compared at multiple control points.

RTG algorithm. The RTG algorithm requires three input parameters: two or more spatiotemporal points, speed, and the time interval between two consecutive points. The input points used were the points of the down-sampled high-frequency part of the white stork trajectory (see above). The speed parameter was extracted from the empirical distribution of the given sample. The default speed was the median of the sample (0.026 m/s) for most cases. The reason for selecting the median and not the maximum speed of the sample data was that the latter was an order of magnitude higher than the most commonly used speed, and it was detected in less than 2% of the instances. In order to approximate the speed behaviour of the object, when higher speed was required, the value was randomly selected from the accepted range of the distribution, hence the maximum speed was taken into account. The time interval was calculated from the real time difference between two consecutive sample points, and the number of points to be created. For instance, if two input points P_1 and P_3 have a time difference $\Delta t = 30$ min and one intermediate point P_2 must be created, the time interval will be $\Delta t/2 = 15$ min; if two points were required the time interval would be 10 mins.

In total 50 trajectories for each of the 951 pairs of points were created. Figure 13 depicts the full result of the algorithm, whereas Figure 14 is a close-up of parts of it. Once the trajectories were created, a line density algorithm was used to create the density surface. This was done for visualisation purposes, contrary to the evaluation procedure described above, where the trajectory counts were rasterised directly. The bandwidth for the line density algorithm was set to 0.3 degrees, which covers the maximum recorded distance between two consecutive points on the real trajectory. A snippet of the result is presented in Figure 15.

Brownian Bridges. The utilisation distribution (UD) of the Brownian Bridge movement model was computed using the move package in R. The same down-sampled trajectory was used as input. The location error was set to 10 m, equal to the estimated GPS error. The margin used for behavioural change point analysis was set to 21m, and

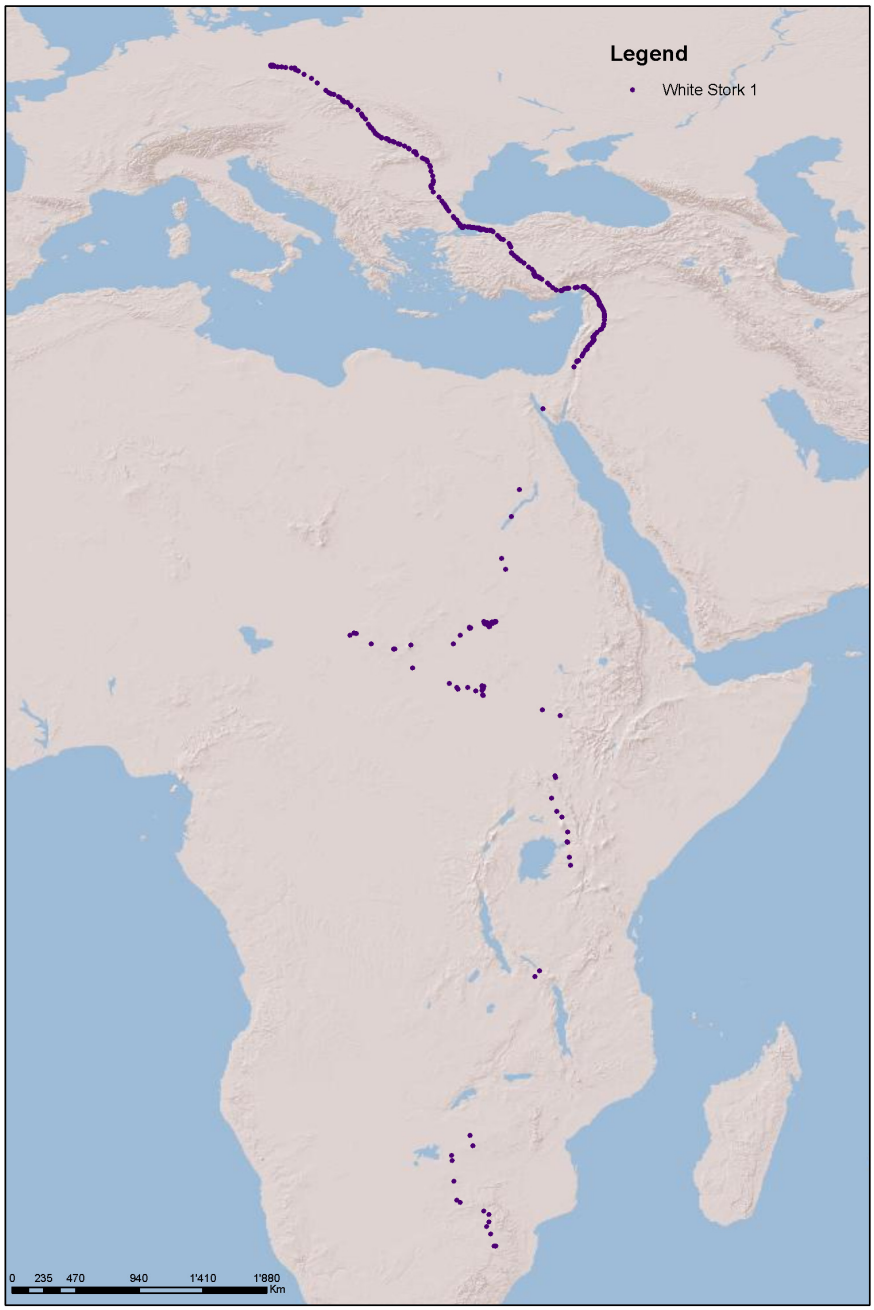

Figure 12.: Overview of the trajectory of White Stork 1

the window size along the track was set to 9m. The calculation extent was set to be 30 % larger than the bounding box of the input points in order to avoid edge effects.

The result of the calculation is depicted in the top row of Figure 16 including the 99 % contour of the UD probability. For a more differentiated picture of the UD variation, the log transformed version is shown to the right.

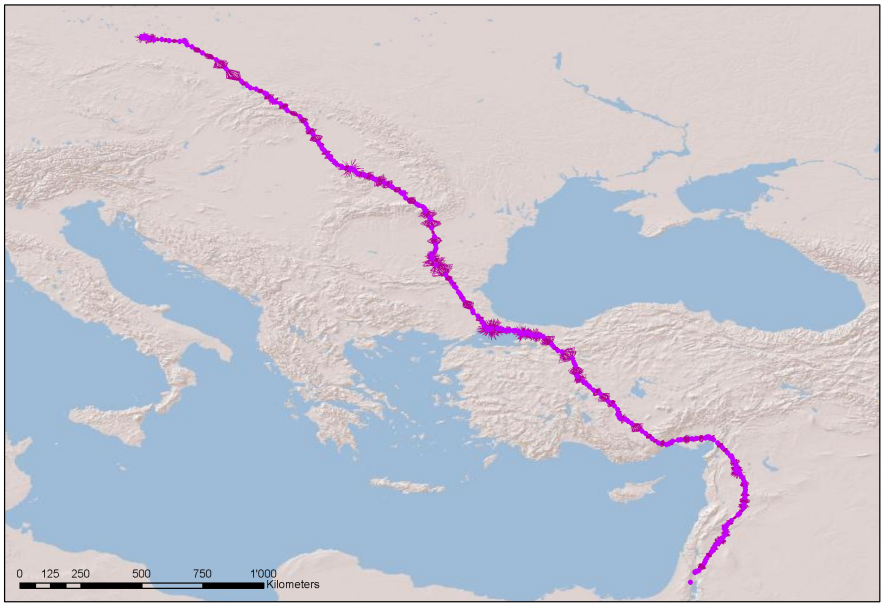

Figure 13.: Resampled and simulated trajectories

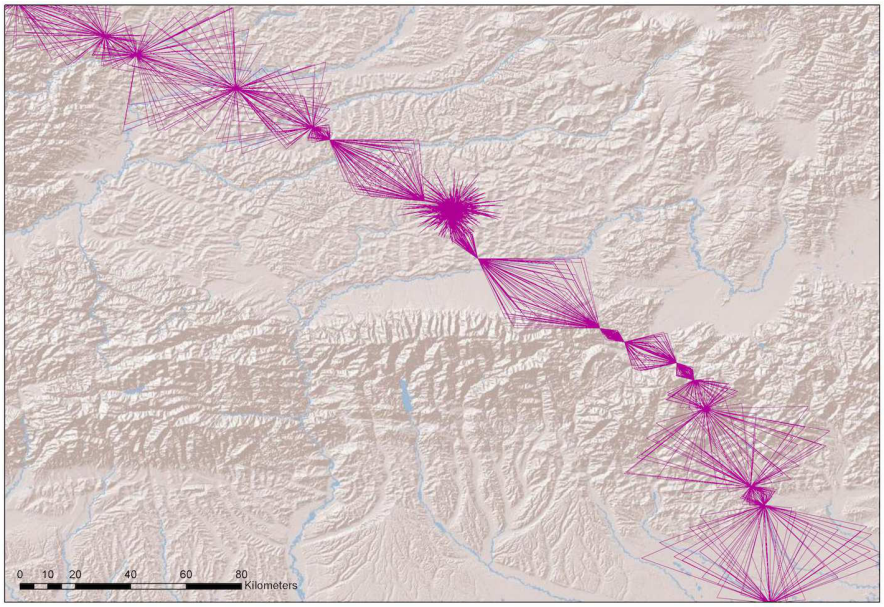

Figure 14.: Close-up view of Figure 13

Comparison. The Brownian bridges method creates a surface of UD probabilities, whereas our RTG algorithm is conditioned to create single random trajectories. At first sight, there seems to be little use in trying to compare these methods. However, both approaches model the movement between an origin and a destination; they both derive from the assumption of random walks; and they both account for the total time of travel.

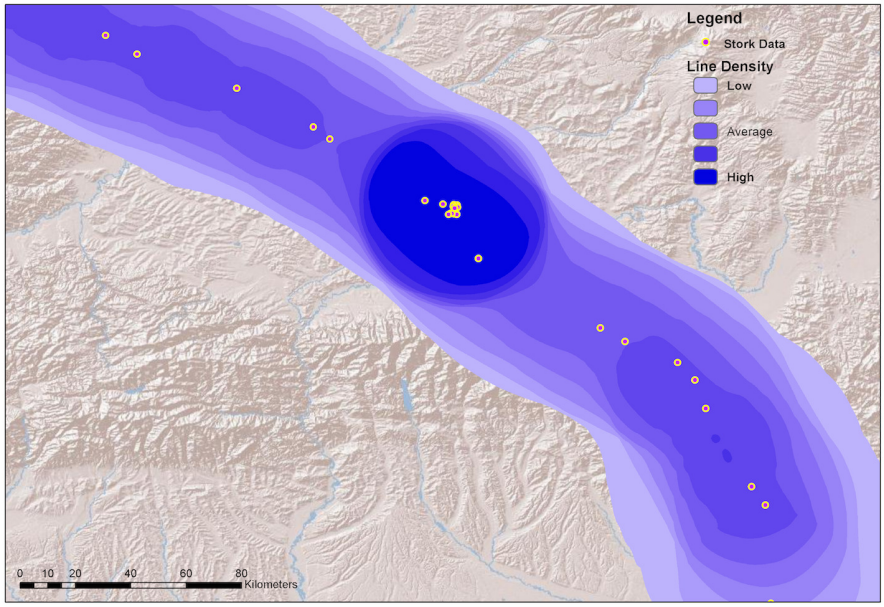

Figure 15.: Line density surface created over the 50 trajectories of Fig. 13

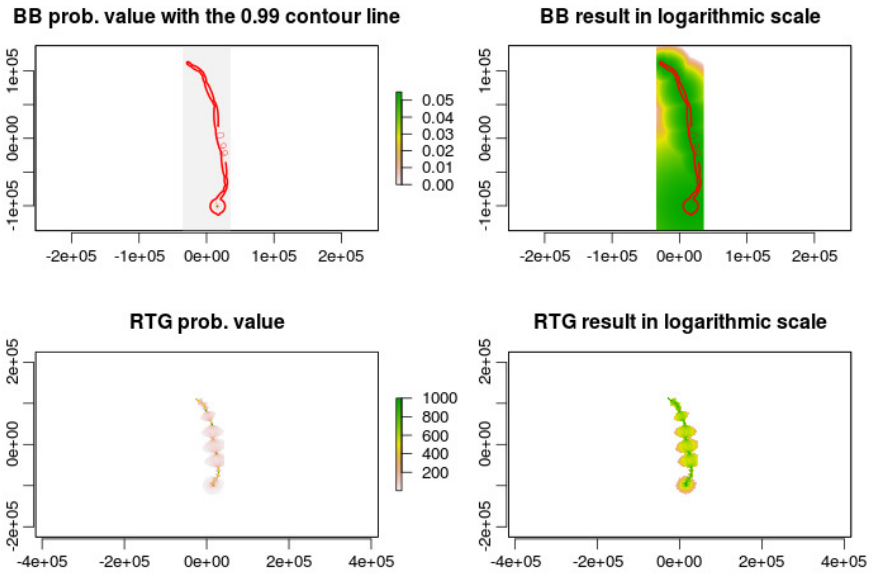

Figure 16.: Top row: Utilisation distribution using the BB movement model, with 99% contour line, in linear (left) and logarithmic (right) scale. Bottom row: Normalised RTG density distribution in linear (left) and logarithmic (right) scale.

For large numbers of trajectories generated with the RTG algorithm, we thus expect to see increasing similarity between the density of RTG trajectories and the corresponding BB occupancy probabilities. That is, while the comparison of the two methods by

absolute numbers is not possible, we may nevertheless compare the trajectory density surface generated by the RTG algorithm with the BB occupancy probability surface, in qualitative terms. For this comparison the RTG trajectories were imported to R, and then rasterised yielding the count of lines within each pixel (Fig. 16). The two rasters (RTG, BB) had identical extent and resolution, and were normalised before the final comparison.

In each of the 951 control point locations we extracted the UD probabilities for both approaches. Figure 17 visualises the results for a random subset of 30 consecutive control points. The plot shows two distinctive clusters in both approaches: low probabilities for the lower half of point IDs; high probabilities for the upper half. These values are more easily explained if we consider the distances between each consecutive pair of points (Fig. 17). The expected value should be 1.0 for all control points. This is achieved quite well at short distances between points, where both simulated scores are almost perfect (i.e. almost reach 1.0). However, the greater the distance between points, the more the two algorithms lose in accuracy, though the RTG algorithm is performing significantly better than the BB. Overall, the average accuracy for the 30 control points was 62.23 % for the RTG algorithm and 48.38 % for BB, where 100 % would imply perfect agreement between the expected and the simulated value.

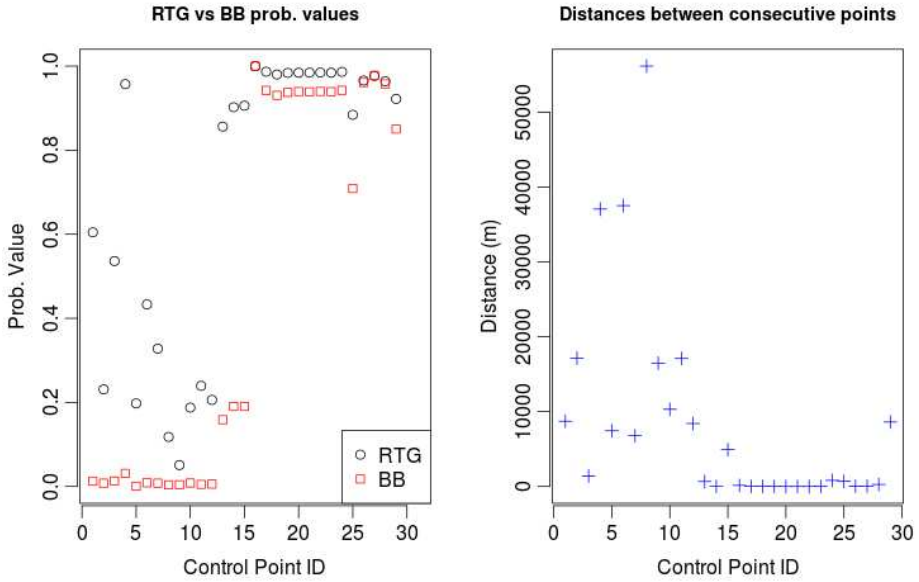

Figure 17.: Left: The probability values for the RTG and BB approach, respectively, extracted at a random subset of 30 consecutive control points. Right: The distances between each pair of consecutive control points

6.3. *Discussion*

In order to draw the overall conclusions from the comparative evaluation of our proposed algorithm, let us start by recalling the requirements stated in Section 2. The task was to develop an algorithm that could generate synthetic trajectories between two given

points. These trajectories should be based on the principle of random walks, and at the same time conditioned by control parameters such as time budget and maximum speed. The algorithm should be efficient, in order to enable the creation of large numbers of trajectories. And it should be able to cope with the spherical shape of the Earth, as the given endpoints may be separated by hundreds or even thousands of kilometres.

The proposed RTG algorithm clearly outperforms other random walk algorithms. As the above empirical evaluation results have shown the RTG algorithm maintains the random walk model basis, with the advantage of reliably and efficiently generating random walks between two given endpoints (i.e. the trajectory is guaranteed to reach the destination). Additionally, the RTG algorithm allows to control the time budget, as opposed to other random walk approaches.

The comparison to Brownian bridges has demonstrated that by generating large numbers of trajectories and creating a density surface out of these, the RTG algorithm produces comparable results. This is not so surprising, since both approaches derive from the random walk model though it should be acknowledged that the distribution used in RTG to generate vertices within the bounding box is a uniform distribution, in contrast to Brownian bridges, where a normal distribution is used perpendicularly to the segment connecting the two end points. However, note that due to the constrained travel time between the two end points, the empirically generated trajectories are actually not uniformly distributed in space but rather follow a distribution that is not unlike that of the Brownian bridge. In addition to that, RTG is free of three shortcomings of the BB method. First, the initial outcome is a trajectory and not a probability surface, allowing to add patterns of specific behaviour in the simulation. Second, in the case of BB the value of probability does not fully decay to 0 (i.e., each pixel within the study area is assigned a non-zero probability). That makes the violation of the maximum speed inevitable in order for the moving object to make it to the destination. Third, as the size of the study area becomes larger and the resolution of the analysis becomes finer, the performance of the BB algorithm decreases dramatically as a function of the raster surface that must be generated. This overhead is particularly large if the aim is only to extract individual trajectories from the BB surface. Conversely, the performance of our RTG algorithm depends linearly on the number of intermediate points that need to be placed. This number, in turn, is dictated only by the time budget, the temporal sampling interval, and the placement of individual points (which is free of complicated numerical simulations, as shown above). Finally, the two approaches could be considered complementary, as RTG can be used to enrich the input data for the BB approach.

Compared to the space-time prism approaches, including Song and Miller (2014), the proposed RTG algorithm avoids the necessity of considering time as a continuous parameter. Instead, RTG focuses on generating individual trajectories for specific time slices in the STP model. This enables our algorithm to handle large geographic distances, accounting for spherical distortion. To our knowledge STP equations for the sphere have not been formulated so far.

7. Conclusions

We have presented an algorithm that can efficiently and reliably generate large numbers of trajectories between two given points on the globe. The trajectories follow the random walk model and can thus be used as null hypothesis for applications in animal ecology such as bird migration. They can be conditioned by control parameters, including time

budget and speed of travel. And since the origin and destination may potentially be spaced far apart, we have presented two versions of our random trajectory generator (RTG) algorithm, one for the 2-D Euclidean plane, and one that accounts for the spherical shape of the Earth.

In essence, the proposed algorithm combines elements from the three leading models in movement simulation, the random walk model, the space-time prism model, and the Brownian bridges movement model. As demonstrated experimentally, it combines the advantages of all three models, and thus also outperforms them.

Nevertheless, the presented algorithm is still limited in terms of its degree of realism. In particular, the facilities available to condition the movement model are still limited to only two user-defined parameters, that being the speed and the available time budget of movement. This was done in order to initially concentrate on the main trajectory generator model. More control parameters could of course be imagined, such as the turning angle distribution. Also, interactions between multiple moving individuals, or attraction/avoidance points or areas in the context could be taken into consideration in order to create realistic corridors in multiple disjunct habitat patches. In the future we therefore intend to exploit the modular nature of the model by incorporating context awareness in the simulation and introducing the energy budget as a limiting factor for the movement. Going a step further, latent learning behaviours, change of behaviours and adaptive penalties could be introduced. Last but not least user-defined parameters will be added, to enable the manipulation of the statistical footprint of the simulated trajectories, ultimately allowing for more realistic simulations.

8. Acknowledgements

We are grateful to Andrea Flack of the Max Planck Institute for Ornithology, Radolfszell, Germany, for providing the data used in Section 6.2. The data can be found on the Movebank website: `https://www.movebank.org`. Funding by the Swiss State Secretariat for Education, Research and Innovation (SERI) through project CASIMO (C09.0167) is gratefully acknowledged.

References

Bartumeus, F., *et al.*, 2010. Fishery discards impact on seabird movement patterns at regional scales.. *Current biology : CB*, 20 (3), 215–22.

Benhamou, S., 2006. Detecting an orientation component in animal paths when the preferred direction is individual-dependent. *Ecological Society of America*, 87 (2), 518–528.

Benhamou, S., 2011. Dynamic Approach to Space and Habitat Use Based on Biased Random Bridges. *PLoS ONE*, 6 (1), e14592.

Biro, D., *et al.*, 2007. Pigeons combine compass and landmark guidance in familiar route navigation.. *Proceedings of the National Academy of Sciences of the United States of America*, 104 (18), 7471–6.

Block, B.a., *et al.*, 2011. Tracking apex marine predator movements in a dynamic ocean.. *Nature*, 475 (7354), 86–90.

Börger, L., Dalziel, B.D., and Fryxell, J.M., 2008. Are there general mechanisms of animal

home range behaviour? A review and prospects for future research.. *Ecology letters*, 11 (6), 637–50.

Brinkhoff, T., 2002. A Framework for Generating Network-Based Moving Objects. *GeoInformatica*, 6 (2), 153–180.

Bullard, F., 1991. Estimating the Home Range of an Animal: A Brownian Bridge Approach. Thesis (PhD). University of North Carolina at Chapel Hill.

Codling, E.a., Bearon, R.N., and Thorn, G.J., 2010. Diffusion about the mean drift location in a biased random walk.. *Ecology*, 91 (10), 3106–13.

Codling, E.A., Plank, M.J., and Benhamou, S., 2008. Random walk models in biology.. *Journal of the Royal Society, Interface / the Royal Society*, 5 (25), 813–34.

Elveback, L.R., *et al.*, 1976. An Influnza simulation model for immunization studies. *American Journal of Epidemiology*, 103 (2), 152–165.

Esa, S., ed. , 1982. *Conceptual Issues in Ecology*. Springer.

Hägerstrand, T., 1970. What about People in Regional Science?. *Papers of the Regional Science Association*, 14, 7–21.

Harris, K.J. and Blackwell, P.G., 2013. Flexible continuous-time modelling for heterogeneous animal movement. *Ecological Modelling*, 255, 29–37.

Horne, J.S., *et al.*, 2007. Analyzing animal movements using Brownian bridges.. *Ecology*, 88 (9), 2354–63.

Humphries, N.E., *et al.*, 2010. Environmental context explains Lévy and Brownian movement patterns of marine predators.. *Nature*, 465 (7301), 1066–9.

Joubert, W., Fourie, P., and Axhausen, K., 2010. Large-Scale Agent-Based Combined Traffic Simulation of Private Cars and Commercial Vehicles. *Transportation Research Record*, 2168, 24–32.

Kuijpers, B., Grimson, R., and Othman, W., 2011. An analytic solution to the alibi query in the space–time prisms model for moving object data. *International Journal of Geographical Information Science*, 25 (2), 293–322.

Kuijpers, B., *et al.*, 2010. Anchor uncertainty and space-time prisms on road networks. *International Journal of Geographical Information Science*, 24 (8), 1223–1248.

Lohmann, K.J., *et al.*, 2004. Geomagnetic map used in sea-turtle navigation. *Nature*, 428 (April), 909–910.

Miller, H.J., 1991. Modelling accessibility using space-time prism concepts within geographical information systems. *International Journal of Geographical Information Systems*, 5 (3), 287–301.

Miller, H.J., 1999. Measuring Space-Time Accessibility Benefits within Transportation Networks: Basic Theory and Computational Procedures. *Geographical Analysis*, 31 (1), 1–26.

Nathan, R., *et al.*, 2008. A movement ecology paradigm for unifying organismal movement research. *PNAS*, 105 (49), 19052–19059.

Pelekis, N., *et al.*, 2013. Hermoupolis: A Trajectory Generator for Simulating Generalized Mobility Patterns. *In*: *Machine Learning and Knowledge Discovery in Databases*, Vol. 8190 of *Lecture Notes in Computer Science Volume* Springer-Verlag, 659–662.

Perony, N., *et al.*, 2012. How random is social behaviour? Disentangling social complexity through the study of a wild house mouse population.. *PLoS computational biology*, 8 (11), e1002786.

Song, Y. and Miller, H.J., 2014. Simulating visit probability distributions within planar space-time prisms. *International Journal of Geographical Information*, 28 (1).

Technitis, G. and Weibel, R., 2012. A Hybrid Geospatial Simulation Model for Moving Objects. *In*: *Proceedings, AutoCarto 2012* Cartography and Geographic Information

Society, 1–16.

Torrens, P.M., *et al.*, 2012. An extensible simulation environment and movement metrics for testing walking behavior in agent-based models. *Computers, Environment and Urban Systems*, 36 (1), 1–17.

Wentz, E.A., Campell, A.F., and Houston, R., 2003. A comparison of two methods to create tracks of moving objects: linear weighted distance and constrained random walk. *International Journal of Geographical Information Science*, 17 (7), 623–645.